CSM 61B Spring 2019

ADTs, Trees

Mentoring 6: March 4, 2019

1 Abstract Data Types

A **list** is an ordered sequence of items: like an array, but without worrying about the length or size.

```
interface List<E> {
    boolean add(E element);
    void add(int index, E element);
    E get(int index);
    int size();
}
```

A **set** is an unordered collection of unique elements.

```
interface Set<E> {
    boolean add(E element);
    boolean contains(Object object);
    int size();
    boolean remove(Object object);
}
```

A **map** is a collection of key-value mappings, like a dictionary in Python. Like a set, the keys in a map are unique.

```
interface Map<K,V> {
    V put(K key, V value);
    V get(K key);
    boolean containsKey(Object key);
    Set<K> keySet();
}
```

2 Interview Questions

2.1 Define a procedure, `sumUp`, which returns **`true`** if any two values in the `array` sum up to `n`.

```
public static boolean sumUp(int[] array, int n) {









}
```

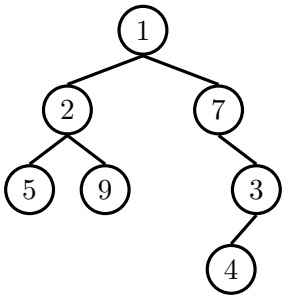
2.2 Define a procedure, `isPermutation`, which returns **`true`** if a string `s1` is a permutation of `s2`. For example, `"atc"` and `"tac"` are permutations of `"cat"`.

```
public static boolean isPermutation(String s1, String s2) {




}
```

3 Binary Trees

3.1 Define a procedure, `height`, which takes in a `Node` and outputs the height of the tree. Recall that the height of a leaf node is 0.

```
public class BinaryTree<T> {
    protected Node root;
    protected class Node {
        public T value;
        public Node left;
        public Node right;
    }
}
```




```
private int height(Node node) {




}
```

What is the runtime of `height`?

3.2 Define a procedure, `isBalanced`, which takes a `Node` and outputs whether or not the tree is balanced. A tree is **balanced** if the left and right branches differ in height by at most one and are themselves balanced.

```
private boolean isBalanced(Node node) {




}
```

What is the runtime of `isBalanced`?

3.3 Define `isSymmetric` which checks whether the binary tree is a mirror of itself.

```
public boolean isSymmetric() {
    if (root == null) {
        return true;
    }
    return isSymmetric(root.left, root.right); // use helper method
}

private boolean isSymmetric(Node left, Node right) {









}
```

4 Binary Search Trees

4.1 Provide tight asymptotic runtime bounds in terms of N, the number of nodes in the tree, for the following operations and data structures.

Operations	Binary Search	Balanced Search
`boolean contains(E e);`		
`boolean add(E e);`		